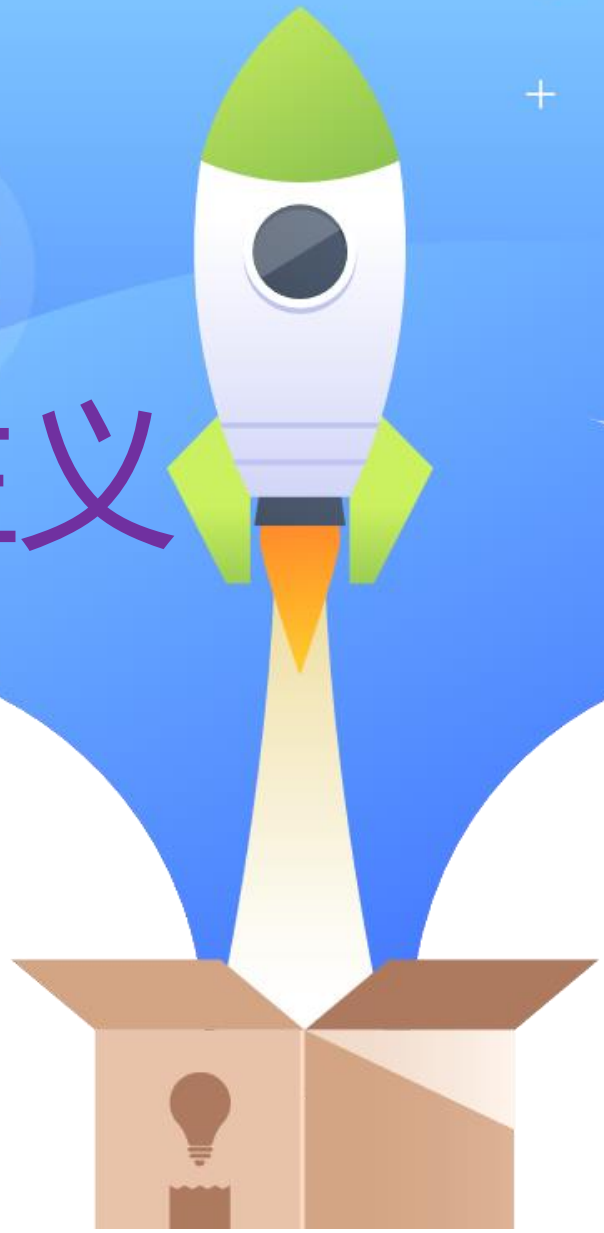
个人主义vs集体主义



圣则IT赋能学院

编程中级课

目　录

CONTENTS

① 温故知新

② 更多字符串运算

③ 字符串内建方法

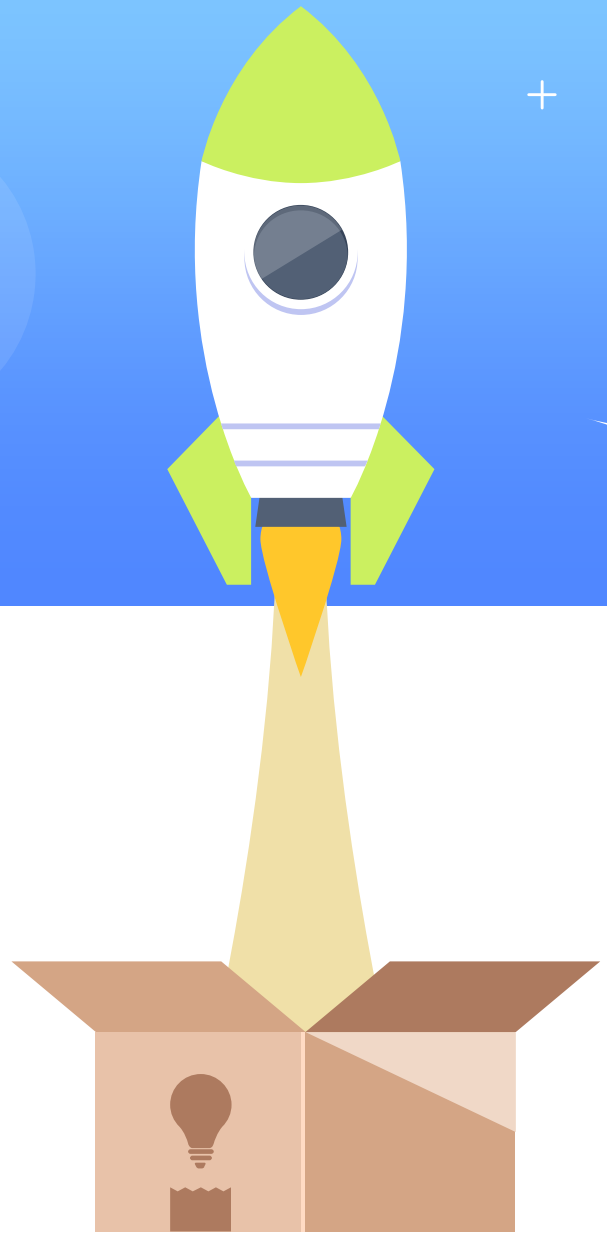
④ 文本文件读取

⑤ 布置作业

⑥ 参考资料

/01

温故知新



字符串

一串字符（字母，数字，符号，空格等）

字符串用单引号 ' 或双引号 " 括起来

```
"A"
"2"
'*'
"A2"
'A+'
'我的成绩是 A+'
""
```

'hello' 等同于 "hello"

合并字符串

```
>>> "Hello" + "World"
'HelloWorld'
>>> "Hello" + " " + "World"
'Hello World'
>>> "bye" + "bye"
'byebye'
>>> text1 = "Hello"
>>> text1 + " Mike"
'Hello Mike'
```

运算符	表达式	意思
+	x + y	合并 x 字符串和 y 字符串

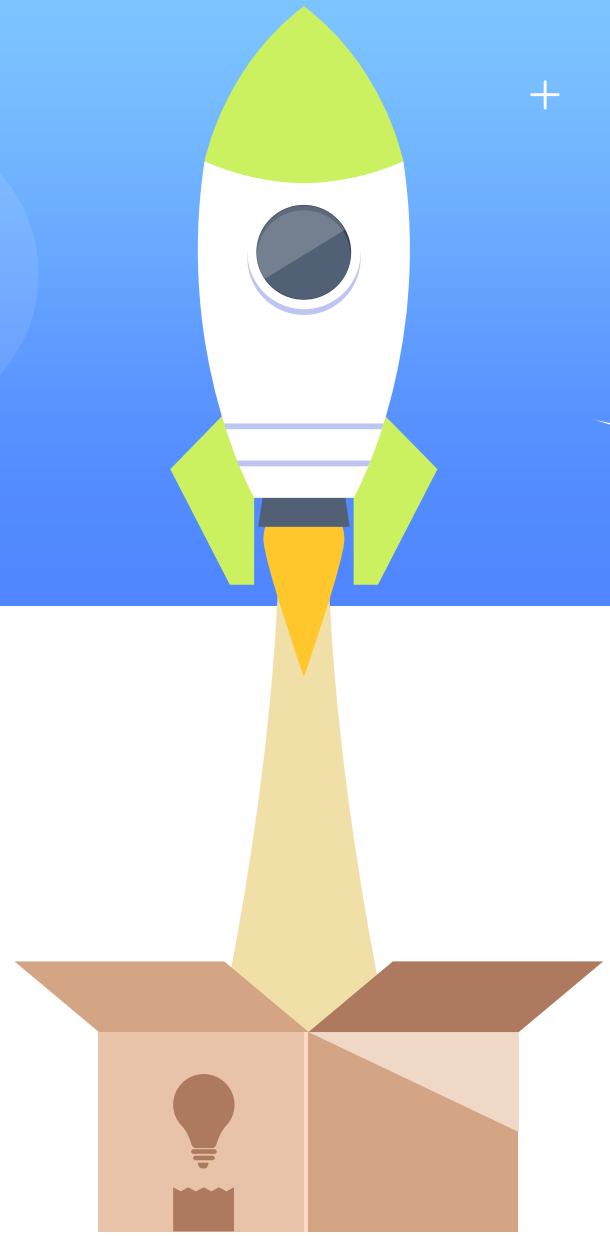
+ 把两个字符串合并成一个新的字符串

包含+的赋值

```
>>> text1 = "Hello"
>>> text2 = text1 + "!"  ⟹ "Hello" + "!"  ⟹ "Hello!"
>>> text2
'Hello!'
>>> text1 = "bye"
>>> text2 = text1 + text1  ⟹ "bye" + "bye"  ⟹ "byebye"
>>> text2
'byebye'
>>> text3 = "hi"
>>> text3 = text3 + text3  ⟹ "hi" + "hi"  ⟹ "hihi"
>>> text3
'hihi'
>>> text4 = text3 + 2
…TypeError: must be str, not int
>>> text4 = text3 + '2'
>>> text4
'hihi2'
```

/02

更多字符串运算



重复字符串

```
>>> text1 = "hi"
>>> text2 = text1 * 2 ⇒ "hi" * 2 ⇒ "hihi"
>>> text2
'hihi'
>>> text1 = "bye"
>>> text2 = 3 * text1 ⇒ 3 * "bye" ⇒ "byebyebye"
>>> text2
'byebyebye'
>>> text3 = "hi"
>>> name = "Mike"
>>> msg = text3 * 2 + ", "+ name + "!"
>>> msg
'hihi, Mike!'
```

运算符	表达式	意思
*	x * n	重复 x n 次

字符串　整数

长度

```
>>> len("Mike")
4
>>> len("123")
3
>>> letters = "ABCDEFG"
>>> len(letters)
7
>>> letters = "A B C"
>>> len(letters)
5
```

len() 函数返回字符串的长度

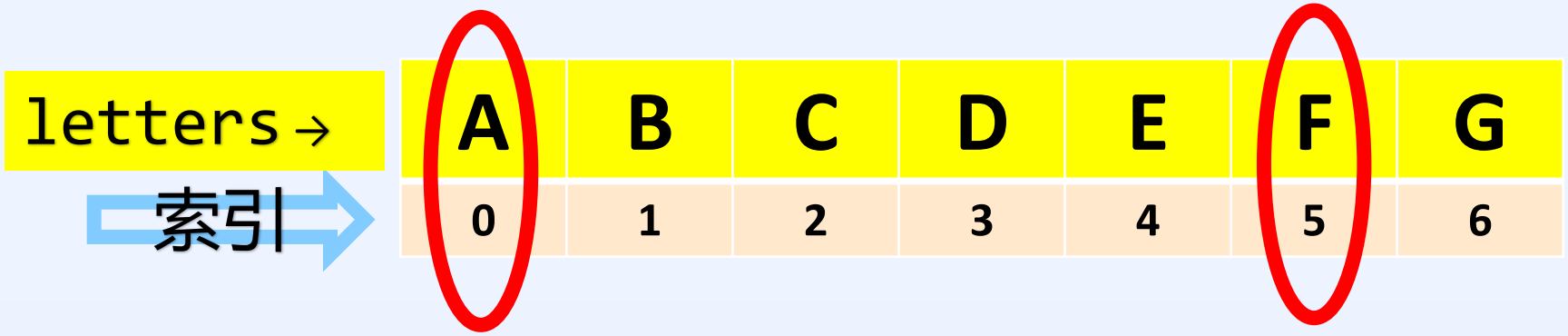
空格也算在内

选择单个字符

```
>>> letters = "ABCDEFG"
>>> letters[0]
'A'
>>> letters[5]
'F'
```

letters[0]从letters选择索引为 0的字符，然后返回一个包含这个字符的新字符串

索引用来选择字符串里的字符
从0开始

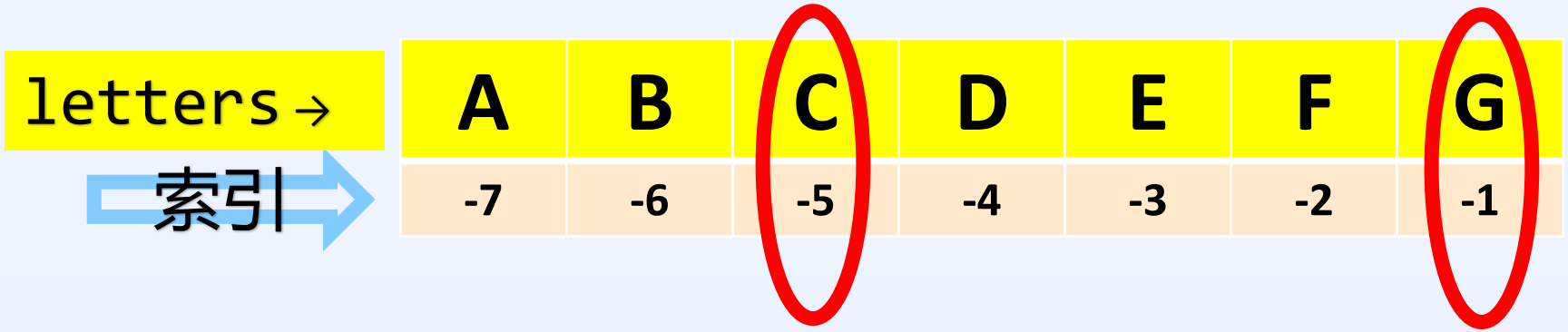


负的索引

```
>>> letters = "ABCDEFG"
>>> letters[-1]
'G'
>>> letters[-5]
'C'
```

letters[-1]从letters选择最后一个字符，然后返回一个包含这个字符的新字符串

负的索引: 从字符串的最后向前数
从 -1 开始



举例

```
>>> text = "Hello World!"
>>> text[0]
'H'
>>> text[5]
' '
>>> text2 = text[0]+text[1]+text[8]
>>> text2
'Her'
>>> text[4] == text[-5]
True
>>> text3 = text[-2]+text[1]+text[-3]
>>> text3
'del'
```

H	e	l	l	o		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

从前面索引

从后面索引

截取部分字符串（裁切）

格式：string[i:j]

返回 **string** 从索引 **i** 到索引 **j**（不包含）的字符

```
>>> letters = "ABCDEFG"
>>> letters[2:5]
'CDE'
>>> letters[4:-1]
'EF'
>>> letters[2:]
'CDEFG'
>>> letters[:4]
'ABCD'
>>> letters[:]
'ABCDEFG'
```

A	B	C	D	E	F	G
0	1	2	3	4	5	6
-7	-6	-5	-4	-3	-2	-1

返回从索引 2 到最后的字符

返回从开始到索引 3的字符

更多例子

```
>>> text = "Hello World!"
>>> text[6:11]
'World'
>>> text[6:-1]
'World'
>>> text2 = text[-6:-1]
>>> text2
'World'
>>> text[6:]
'World!'
>>> text[:5] + text[-1]
'Hello!'
```

H	e	l	l	o		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

跳跃式截取/裁切

格式：string[i:j:k]

返回 string 从索引 i 到索引 j（不包含）的字符，索引差数为 **k**

```
>>> letters = "ABCDEFG"
>>> letters[1:6]
'BCDEF'
>>> letters[1:6:1]
'BCDEF'
>>> letters[1:6:2]
'BDF'
>>> letters[6:1:-1]
'GFEDC'
>>> letters[1:6:-1]
''
```

A	B	C	D	E	F	G
0	1	2	3	4	5	6
-7	-6	-5	-4	-3	-2	-1

选择索引为1的字符，然后加 2, …
直到索引为5 5

从后向前选择

更多例子

```
>>> text = "ABCDEFGHIJK"
>>> text[-1:-6:-2]
'KIG'
>>> text[-6:-1:-2]
''
>>> text[5::-2]
'FDB'
>>> text[5::2]
'FHJ'
>>> text[:-6:-3]
'KH'
>>> text[:-6:3]
'AD'
```

当差数为负数时，默认值如下：
开始索引 -> 最后一个字符
结束索引 -> 第一个字符

A	B	C	D	E	F	G	H	I	J	K
0	1	2	3	4	5	6	7	8	9	10
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

成员运算符

```
>>> "b" in "banana"
True
>>> "o" in "banana"
False
>>> "an" in "banana"
True
>>> "pan" in "banana"
False
>>> "banana" in "banana"
True
>>> "b" not in "banana"
False
>>> "o" not in "banana"
True
>>> "an" not in "banana"
False
```

如果右边的字符串包含左边的字符串，返回True

一个字符串包含它自己

not in 和 in 相反

圣经人物猜猜看

- 你的朋友读经时，读到了一个圣经人物，他只告诉你名字有几个字，希望你可以猜一下。
- 试着猜名字或其中的一部分
- 如果猜对了，你的朋友会说：猜对啦！
- 否则，他会说： 下次再试试！

用程序来模拟

1. 定义一个变量，用来储存圣经人物的名字
2. 显示名字有多少个字符
3. 提示用户输入他猜的名字或一部分
4. 检测输入的字符串是否是名字的一部分，并显示相应的结果

len()

input()

in
if/else

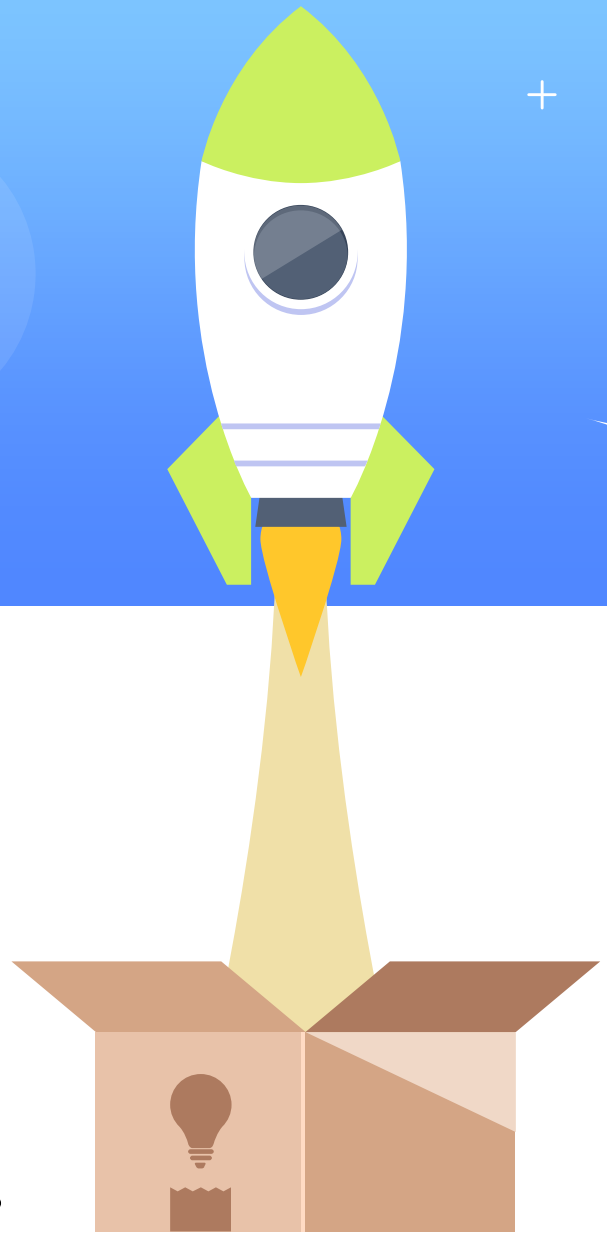
方法1

```
name = "亚伯拉罕"
print("名字的长度是", len(name))
g = input("请输入你猜到的名字或其中的一部分: ")
if g in name:
    print("猜对了!")
else:
    print("下次再试试!")
```

分组练习和互动
(15-20分钟)

/03

字符串内建方法



注意：所有字符串方法都返回新值。它们不会更改原始字符串。

原材料

咖啡机

产物

执行一个具体的任务：
泡咖啡

输入

参数

函数
`make_coffee()`

输出

返回值

咖啡机 1

c1

咖啡机2

c2

如何用咖啡机 1 来制作咖啡?

```
c1.make_coffee(参数)
```

三类方法

- 检查：是否满足某个条件（True/False）
  - isupper(), islower(), isalpha(), isdigit()
- 转换：生成新的字符串或。。。
  - upper(), lower()
- 搜索：得到相关的信息
  - find(), count()

upper()

```
>>> "a".upper()
"A"
>>> "ab".upper()
"AB"
>>> "a2".upper()
"A2"
>>> "hi, mike!".upper()
"HI, MIKE!"
>>> s = "apple"
>>> s.upper()
"APPLE"
```

upper() 方法返回大写的字符串

非字母的字符保持不变

lower()

```
>>> "A".lower()
"a"
>>> "AB".lower()
"ab"
>>> "A2".lower()
"a2"
>>> "Hi, Mike!".lower()
"hi, mike!"
>>> s = "Apple Pie"
>>> s.lower()
"apple pie"
```

lower() 返回小写的字符串

非字母的字符保持不变

isupper()

```
>>> "A".isupper()
True
>>> "ABC".isupper()
True
>>> "a".isupper()
False
>>> "Ab".isupper()
False
>>> "".isupper()
False
>>> "A2".isupper()
True
>>> "YEAH!".isupper()
True
>>> s = "YEAH!"
>>> s.isupper()
True
```

如果字符串中至少有一个字母，而且的所有字符（字母）都是**大写**，则返回 True，否则返回 False。

字符串中可以包含非字母字符

islower()

```
>>> "a".islower()
True
>>> "abc".islower()
True
>>> "A".islower()
False
>>> "Ab".islower()
False
>>> "".islower()
False
>>> "a2".islower()
True
>>> "yeah!".islower()
True
>>> s = "yeah!"
>>> s.islower()
True
```

如果字符串中至少有一个字母，而且的所有字符（字母）都是**小写**，则返回 True，否则返回 False。

字符串中可以包含非字母字符

isalpha()

```
>>> "A".isalpha()
True
>>> "A2".isalpha()
False
>>> "A ".isalpha()
False
>>> "".isalpha()
False
>>> "Mike!".isalpha()
False
>>> s = "Apple Pie"
>>> s.isalpha()
False
```

如果字符串至少有一个字符，而且其中的所有字符都是字母，则返回 True，否则返回 False。

isdigit()

```
>>> "2".isdigit()
True
>>> "0123".isdigit()
True
>>> "A2".isdigit()
False
>>> "2 ".isdigit()
False
>>> "".isdigit()
False
>>> "100.2".isdigit()
False
>>> s = "100"
>>> s.isdigit()
True
>>> s = "100+2"
>>> s.isdigit()
False
```

如果字符串中的所有字符都是数字，则返回 True，否则返回 False。

find()

```
>>> "banana".find("b")
0
>>> "banana".find("a")
1
>>> "banana".find("c")
-1
>>> "banana".find("an")
1
>>> s = "apple"
>>> s.find("l")
3
>>> s.find("p")
1
>>> s.find("pp")
1
```

在字符串中搜索指定的值并返回它被找到的位置。

如果找不到，返回 -1

count()

```
>>> "banana".count("b")
1
>>> "banana".count("a")
3
>>> "banana".count("c")
0
>>> "banana".count("an")
2
>>> s = "apple"
>>> s.count("l")
1
>>> s.count("p")
2
>>> s.count("pp")
1
```

返回指定值在字符串中出现的次数。

方法vs函数

和函数类似，只不过要跟**对象**联系在一起

make_coffee() 只有咖啡机对象才可以调用

字符串方法 => 字符串才可以调用的

格式:

```
字符串.方法名(参数)
```

举例:

```
"banana".find("b")
```

参数

0, 1 或更多，取决于方法的定义

举例:

```
"banana".find("b")
```

1个参数

```
"apple".upper()
```

0个参数

圣经人物猜猜看

- 你的朋友读经时，读到了一个圣经人物，他只告诉你名字有几个字，希望你可以猜一下。
- 试着猜名字或其中的一部分
- 如果猜对了，你的朋友会说：猜对啦！
- 否则，他会说： 下次再试试！

用程序来模拟

1. 定义一个变量，用来储存圣经人物的名字
2. 显示名字有多少个字符
3. 提示用户输入他猜的名字或一部分
4. 检测输入的字符串是否是名字的一部分，并显示相应的结果

len()

input()

in
if/else

方法1

```
name = "亚伯拉罕"
print("名字的长度是", len(name))
g = input("请输入你猜到的名字或其中的一部分: ")
if g in name:
    print("猜对了!")
else:
    print("下次再试试!")
```

除了用 "in"，还有其他的方法吗?

方法2

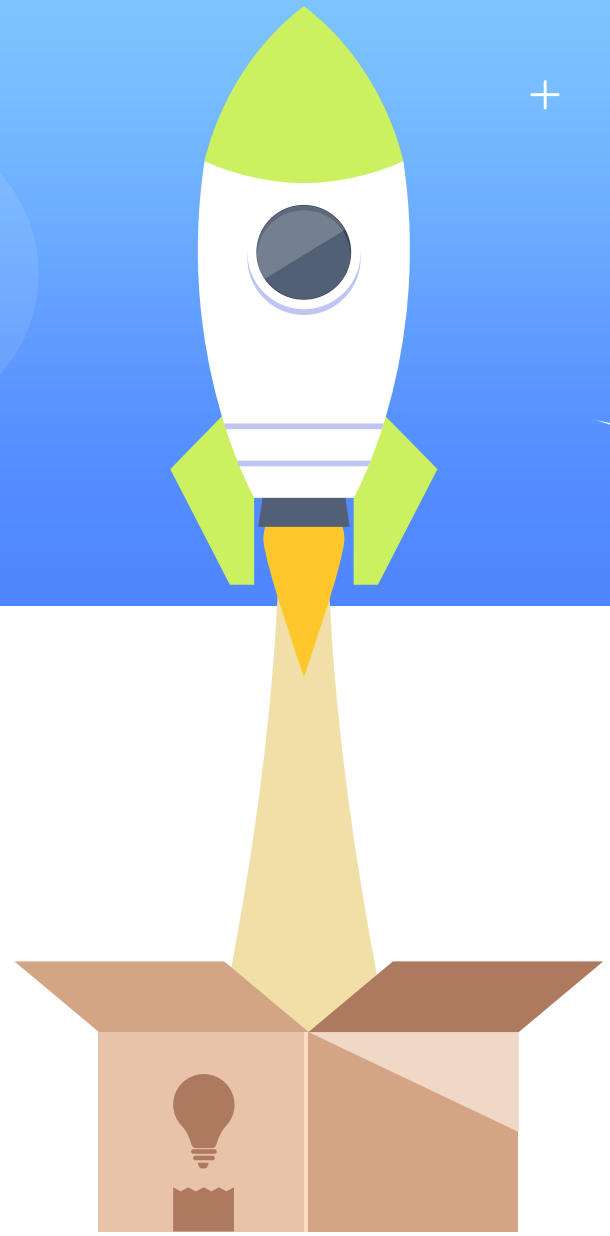
```
name = "亚伯拉罕"
print("名字的长度是", len(name))
g = input("请输入你猜到的名字或其中的一部分: ")
p = name.find(g)
if p != -1 :
    print("猜对了!")
else:
    print("下次再试试!")
```

```
if name.find(g) != -1:
```

分组练习和互动
(10-15分钟)

/04

文本文件读取



步骤

demo1.txt

```
Hello!
Welcome to join the intermediate programming course!
Wish you all a pleasant learning experience!
```

```
f = open("demo1.txt")
c = f.read()
print(c)
```

1. 打开文件

2. 读取文件

3. 后期处理

步骤

demo2.txt

大家好!
欢迎加入编程中级班!
希望大家学习愉快!

字符解码

```
f = open("demo2.txt", encoding="UTF8")
c = f.read()
print(c)
```

参考资料

- 字符串 https://www.w3school.com.cn/python/python_strings.asp
- 文件打开 https://www.w3school.com.cn/python/python_file_handling.asp
- 文件读取 https://www.w3school.com.cn/python/python_file_open.asp